Unit 1: Creative Computing: Building with Blocks

Content Area: **Template**
Course(s):
Time Period: **Full Year**
Length: **Full Year**
Status: **Published**

UNIT RATIONALE

Unit 1 welcomes new and returning students to the world of computer science and coding fundamentals. Students work with MIT App Inventor to create basic apps that rely on the concepts of event-driven programming, branching, iteration, variables, and abstraction—the building blocks of creating with code. Students are introduced to essential computational thinking practices, such as developing abstractions, collaborating around computing, and communicating as they create, test, and refine computational artifacts of Android™ apps.

ESSENTIAL QUESTIONS

What information is being hidden or abstracted by a program?

Why are user stories and user-centered design so important when creating an app?

What are the advantages and challenges of pair programming?

What does it mean for data to persist?

Why are APIs such an essential tool in computer science today?

Why is sharing code and looking at many examples important to people writing programs?

What are some advantages to programming in a text-based language compared to a block-based programming language?

STANDARDS

NEW JERSEY STUDENT LEARNING STANDARDS: CONTENT AREA

New Jersey Core Curriculum - Grade 9 - Technology

8.2.12.B.4

Investigate a technology used in a given period of history, e.g., stone age, industrial revolution or information age, and identify their impact and how they may have changed to meet human needs and wants.

New Jersey (NJSLS) - Grades 9-12 - Computer Science and Design Thinking (2020)

8.1.12.CS.1:

Describe ways in which integrated systems hide underlying implementation details to simplify user experiences.

8.1.12.CS.2:

Model interactions between application software, system software, and hardware.

8.1.12.CS.3:

Compare the functions of application software, system software, and hardware.

8.1.12.CS.4:

Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.

8.1.12.NI.2:

Evaluate security measures to address various common security threats.

8.1.12.NI.3:

Explain how the needs of users and the sensitivity of data determine the level of security implemented.

8.1.12.IC.1:

Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices.

8.1.12.IC.2:

Test and refine computational artifacts to reduce bias and equity deficits.

8.1.12.IC.3:

Predict the potential impacts and implications of emerging technologies on larger social, economic, and political structures, using evidence from credible sources.

8.1.12.DA.2:

Describe the trade-offs in how and where data is organized and stored.

8.1.12.DA.5:

Create data visualizations from large data sets to summarize, communicate, and support different interpretations of real-world phenomena.

8.1.12.AP.1:

Design algorithms to solve computational problems using a combination of original and existing algorithms.

8.1.12.AP.2:

Create generalized computational solutions using collections instead of repeatedly using simple variables.

8.1.12.AP.3:

Select and combine control structures for a specific application based upon performance and readability, and identify trade-offs to justify the choice.

8.1.12.AP.4:

Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue.

8.1.12.AP.5:

Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.

8.1.12.AP.6:

Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.

8.1.12.AP.7:

Collaboratively design and develop programs and artifacts for broad audiences by incorporating feedback from

users.

8.1.12.AP.8:

Evaluate and refine computational artifacts to make them more usable and accessible.

8.1.12.AP.9:

Collaboratively document and present design decisions in the development of complex programs.

8.2.12.ED.5:

Evaluate the effectiveness of a product or system based on factors that are related to its requirements, specifications, and constraints (e.g., safety, reliability, economic considerations, quality control, environmental concerns, manufacturability, maintenance and repair, ergonomics).

8.2.12.NT.1:

Explain how different groups can contribute to the overall design of a product.

CS.9-12.8.1.12.AP.1	Design algorithms to solve computational problems using a combination of original and existing algorithms.
CS.9-12.8.1.12.AP.2	Create generalized computational solutions using collections instead of repeatedly using simple variables.
CS.9-12.8.1.12.AP.3	Select and combine control structures for a specific application based upon performance and readability, and identify trade-offs to justify the choice.
CS.9-12.8.1.12.AP.4	Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue.
CS.9-12.8.1.12.AP.5	Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
CS.9-12.8.1.12.AP.6	Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.
CS.9-12.8.1.12.AP.7	Collaboratively design and develop programs and artifacts for broad audiences by incorporating feedback from users.
CS.9-12.8.1.12.AP.8	Evaluate and refine computational artifacts to make them more usable and accessible.
CS.9-12.8.1.12.AP.9	Collaboratively document and present design decisions in the development of complex programs.
CS.9-12.8.1.12.CS.1	Describe ways in which integrated systems hide underlying implementation details to simplify user experiences.
CS.9-12.8.1.12.CS.2	Model interactions between application software, system software, and hardware.
CS.9-12.8.1.12.CS.3	Compare the functions of application software, system software, and hardware.
CS.9-12.8.1.12.CS.4	Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.
CS.9-12.8.1.12.DA.2	Describe the trade-offs in how and where data is organized and stored.
CS.9-12.8.1.12.DA.5	Create data visualizations from large data sets to summarize, communicate, and support different interpretations of real-world phenomena.
CS.9-12.8.1.12.IC.1	Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices.
CS.9-12.8.1.12.IC.2	Test and refine computational artifacts to reduce bias and equity deficits.
CS.9-12.8.1.12.IC.3	Predict the potential impacts and implications of emerging technologies on larger social, economic, and political structures, using evidence from credible sources.
CS.9-12.8.1.12.NI.2	Evaluate security measures to address various common security threats.

CS.9-12.8.1.12.NI.3	Explain how the needs of users and the sensitivity of data determine the level of security implemented.
CS.9-12.8.2.12.ED.5	Evaluate the effectiveness of a product or system based on factors that are related to its requirements, specifications, and constraints (e.g., safety, reliability, economic considerations, quality control, environmental concerns, manufacturability, maintenance and repair, ergonomics).
CS.9-12.8.2.12.NT.1	Explain how different groups can contribute to the overall design of a product.
TECH.8.2.12.B.4	Investigate a technology used in a given period of history, e.g., stone age, industrial revolution or information age, and identify their impact and how they may have changed to meet human needs and wants.

NEW JERSEY Learning Standards: 21st Century

NEW JERSEY STUDENT LEARNING STANDARDS: CAREER READINESS, LIFE LITERACIES AND KEY SKILLS

12.9.3.ST.2	Use technology to acquire, manipulate, analyze and report data.
12.9.3.ST.6	Demonstrate technical skills needed in a chosen STEM field.
12.9.3.IT-PRG.2	Demonstrate the use of industry standard strategies and project planning to meet customer specifications.
12.9.3.IT-PRG.6	Program a computer application using the appropriate programming language.
12.9.3.IT-PRG.7	Demonstrate software testing procedures to ensure quality products.
12.9.3.LW-SEC.1	Demonstrate effective communications skills (e.g., writing, speaking, listening and nonverbal communication) when communicating security-related directives, technical concepts and other information.
12.9.3.MN-QA.4	Employ project management processes using data and tools to deliver quality, value-added products.
12.9.3.ST-ET.2	Display and communicate STEM information.
12.9.3.ST-ET.3	Apply processes and concepts for the use of technological tools in STEM.
PFL.9.1.12.A.3	Analyze the relationship between various careers and personal earning goals.
CAEP.9.2.12.C.3	Identify transferable career skills and design alternate career plans.

NEW JERSEY STUDENT LEARNING STANDARDS: COMPUTER SCIENCE AND DESIGN THINKING

See content area standards.

REFLECTIONS

Students must do Activities 1.1.1 to 1.1.4 independently. Pair programming starts with Activity 1.1.5

Use vocabulary (temp chart or vocab notebook) so students are familiar with user stories, etc.

PRE-ASSESSMENTS

Student survey.

Teacher will demonstrate one or two event handlers (in MIT App Inventor) as a review for the first activity, so students can see how to identify the event, select it from the Blocks drawer, and drag the blocks into the viewer. After demonstrating a few event handlers, students will continue on their own with their elbow partners for support.

INSTRUCTIONAL PLAN

MODULE 1

LESSON 1.1

Lesson 1.1 Introduction to Computer Science Essentials

Mobile computing has changed our world, and many of today's students have never known a life without apps. This lesson gives students the tools they need to create their own apps using MIT App Inventor. The goal of this lesson is to introduce students to coding fundamentals through block-based programming. Students will develop independent and collaborative strategies that will help them communicate around computing as they learn and reinforce the fundamental concepts of coding. With a powerful yet approachable tool, students will use their creativity to produce computational artifacts like those that are essential to all of us today.

Activity 1.1.1 Getting Started with Block-Based Programming: Digital Doodle (3 days)
Activity 1.1.2 Algorithms and Coding Fundamentals: Happy Accelerometer (3 days)
Activity 1.1.3 Conditionals and Event-Driven Programming: Happy Balance (2 days)
Activity 1.1.4 Local and Global Variables: Guessing Game 2 Player (3 days)
Activity 1.1.5 Iteration and Loops: Guessing Game 1 Player (3 days)
Project 1.1.6 App Development: Creative Expression (5 days)

12.9.3.MN-QA.4	Employ project management processes using data and tools to deliver quality, value-added products.
CS.9-12.8.1.12.AP.1	Design algorithms to solve computational problems using a combination of original and existing algorithms.
CS.9-12.8.1.12.AP.3	Select and combine control structures for a specific application based upon performance and readability, and identify trade-offs to justify the choice.
CS.9-12.8.1.12.AP.4	Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue.
CS.9-12.8.1.12.AP.5	Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
CS.9-12.8.1.12.AP.6	Create artifacts by using procedures within a program, combinations of data and

	procedures, or independent but interrelated programs.
CS.9-12.8.1.12.AP.8	Evaluate and refine computational artifacts to make them more usable and accessible.
CS.9-12.8.1.12.AP.9	Collaboratively document and present design decisions in the development of complex programs.
CS.9-12.8.1.12.CS.1	Describe ways in which integrated systems hide underlying implementation details to simplify user experiences.
CS.9-12.8.1.12.CS.2	Model interactions between application software, system software, and hardware.
CS.9-12.8.2.12.ED.5	Evaluate the effectiveness of a product or system based on factors that are related to its requirements, specifications, and constraints (e.g., safety, reliability, economic considerations, quality control, environmental concerns, manufacturability, maintenance and repair, ergonomics).

Activity 1.1.1

Student Learning Intentions (SLI) WALT: (We are learning to…)	**Activity 1.1.1** Use block-based programming Get started with MIT App Inventor Develop an app independently for creative expression.
Student Learning Strategies	Journaling TEMP Charts (Term, Example, Meaning, Picture) Collaboration Cooperative Learning Pair Programming APB Approach (Activities, Projects, Problems) Class Discussions
Success Criteria	Working Digital Doodle app
Formative Assessment (drives instructional decisions)	Activities and projects Screenshots of code Vocabulary quizzes Call out responses Conclusion questions
Activities and Resources	Create an app that allows the user to take a picture and then draw on the picture in the user interface.
Suggested Modifications	**English Language Learners** **Adjusted Speech**: The teacher changes speech patterns to increase student comprehension. This could include facing the students, paraphrasing, clearly indicating the most important ideas and speaking more slowly. **Visuals**: The teacher uses graphics, pictures, visuals, and manipulatives. This helps ELL students better understand and comprehend the subjects at hand. **Front-Loading Vocabulary**: The teacher front loads vocabulary. This means providing students with a list of important vocabulary words they will need to know for a book, lesson, etc. prior to the lesson being taught. Including pictures to go with the vocabulary words is also very beneficial for the students. **Students with Individualized Education Plans/504s** **Chunking**: The teacher presents information in a way that makes easy for students to understand and remember. Chunking is base on the presumption that our working memory is easily overloaded excessive detail. The best way to deliver information is to organize into meaningful units. Because students with special needs get overloaded easily, chunking is an effective strategy to use with the **Checking for Understanding**: It is important to constantly check understanding, especially for students who have accommodations Teachers want to make sure students understand the concepts being covered in a way that makes sense to them.

Extra time: The teacher provides students with special needs extr time to complete work or answer questions. It is important to give students enough time to process their thoughts.

Oral Reading: The teacher will read work orally to students. Class work such as tests and literature circles may need to be read alou to the student.

Gifted & Talented Strategies
Extensions/Enrichments: Teachers will provide gifted and talent students with extension/enrichment projects. Students will be challenged to further their understanding, to apply acquired knowledge, and/or to produce something in reference to acquired knowledge.

Modify/Change Activities: Teachers will monitor and modify activities to accommodate those students who need to be challenged further. Additional reading, problem-solving, writing, or project work is necessary for those students who are ready to mov on at a rate more accelerated than their peers. In this way, G & T students are provided the same opportunity for support as special needs students.

Students at Risk of School Failure
Directions or Instructions: Make sure directions and/or instructio are given in limited numbers. Give directions/instructions verbally and in simple written format. Ask students to repeat the instructior or directions to ensure understanding occurs. Check back with the student to ensure he/she hasn't forgotten.

Peer Support: Peers can help build confidence in other students assisting in peer learning. Many teachers use the 'ask 3 before me approach. This is fine, however, a student at risk may have to hav a specific student or two to ask. Set this up for the student so he/s knows who to ask for clarification before going to you.

Alternate or Modified Assignments: Always ask yourself, "How can I modify this assignment to ensure the students at risk are abl to complete it?" Sometimes you'll simplify the task, reduce the len of the assignment or allow for a different mode of delivery. For instance, many students may hand something in, the at-risk stude may jot notes and give you the information verbally. Or, it just may be that you will need to assign an alternate assignment.

Increase One to One Time: When other students are working, always touch base with your students at risk and find out if they're on track or needing some additional support. A few minutes here and there will go a long way to intervene as the need presents itse

Contracts: It helps to have a working contract between you and your students at risk. This helps prioritize the tasks that need to be done and ensure completion happens. Each day write down what needs to be completed, as the tasks are done, provide a checkma or happy face. The goal of using contracts is to eventually have th student come to you for completion sign-offs.

Hands On: As much as possible, think in concrete terms and provide hands-on tasks. This means a child doing math may requi a calculator or counters. The child may need to tape record comprehension activities instead of writing them. A child may have to listen to a story being read instead of reading it him/herself.

Tests/Assessments: Tests can be done orally if need be. Break tests down in smaller increments by having a portion of the test in the morning, another portion after lunch and the final part the next day.

Seating: Seat students near a helping peer or with quick access t

	the teacher. Those with hearing or sight issues need to be close t the instruction which often means near the front.

Activity 1.1.2

Student Learning Intentions (SLI) WALT: (We are learning to…)	**Activity 1.1.2** Learn coding fundamentals Apply file naming conventions and version control Develop and test an app incrementally Develop an app independently for a physical game
Student Learning Strategies	Journaling TEMP Charts (Term, Example, Meaning, Picture) Collaboration Cooperative Learning Pair Programming APB Approach (Activities, Projects, Problems) Class Discussions
Success Criteria	Working Happy Accelerometer app
Formative Assessment (drives instructional decisions)	Activities and projects Screenshots of code Vocabulary quizzes Call out responses Conclusion questions
Activities and Resources	Create an app that allows the user to balance a happy face in the center of the screen using the accelerometer built into the mobile device as input.
Suggested Modifications	See Activity 1.1.1

Activity 1.1.3

Student Learning Intentions (SLI) WALT: (We are learning to…)	**Activity 1.1.3** Learn how to use conditionals to make choices in a program Learn to use modifiers to create chained conditionals Personalize the user interface and features of an app incrementall Apply coding fundamentals to create algorithms
Student Learning Strategies	Journaling TEMP Charts (Term, Example, Meaning, Picture) Collaboration Cooperative Learning Pair Programming APB Approach (Activities, Projects, Problems) Class Discussions
Success Criteria	Working Happy Balance app
Formative Assessment (drives instructional decisions)	Activities and projects Screenshots of code Vocabulary quizzes Call out responses Conclusion questions
Activities and Resources	Modify the app created in the previous activity to make a physical balance game. This will include conditionals that let the player kn when they have hit the edge of the screen, a reset game option a the happy face hits as edge, and a sad face image when an edge hit. Students may also choose to add other modifiers that personalize the app and make the game more challenging.
Suggested Modifications	See Activity 1.1.1

Activity 1.1.4

Student Learning Intentions (SLI) WALT: (We are learning to…)	**Activity 1.1.4** Learn when to use local variables and global variables Write programs as pseudocode and in a natural language
Student Learning Strategies	Journaling TEMP Charts (Term, Example, Meaning, Picture) Collaboration Cooperative Learning Pair Programming APB Approach (Activities, Projects, Problems) Class Discussions
Success Criteria	Working Guessing Game 2 player app
Formative Assessment (drives instructional decisions)	Activities and projects Screenshots of code Vocabulary quizzes Call out responses Conclusion questions
Activities and Resources	Create an app that allows one user to enter a number and a seco user to guess the number.
Suggested Modifications	See Activity 1.1.1

Activity 1.1.5

Student Learning Intentions (SLI) WALT: (We are learning to…)	**Activity 1.1.5** Learn to use while loops in a program Learn to increment a count Apply coding fundamentals to create algorithms
Student Learning Strategies	Journaling TEMP Charts (Term, Example, Meaning, Picture) Collaboration Cooperative Learning Pair Programming APB Approach (Activities, Projects, Problems) Class Discussions
Success Criteria	Working Guessing Game 1 Player app
Formative Assessment (drives instructional decisions)	Activities and projects Screenshots of code Vocabulary quizzes Call out responses Conclusion questions
Activities and Resources	Add a random number generator to modify the app from the previous activity from a two-player game to a one-player game.
Suggested Modifications	See Activity 1.1.1

Project 1.1.6

Student Learning Intentions (SLI) WALT: (We are learning to…)	**Project 1.1.6** Decompose a project into smaller parts Apply coding fundamentals and iterative processes
Student Learning Strategies	Journaling TEMP Charts (Term, Example, Meaning, Picture)

	Collaboration Cooperative Learning Pair Programming APB Approach (Activities, Projects, Problems) Class Discussions
Success Criteria	Working Digital Doodle app
Formative Assessment (drives instructional decisions)	Activities and projects Screenshots of code Vocabulary quizzes Call out responses Conclusion questions
Activities and Resources	Create a game, an interactive story, or an app for artistic expressi
Suggested Modifications	See Activity 1.1.1

LESSON 1.2

Lesson 1.2 Collaborating Around Computing

This lesson focuses on collaborative strategies that coding professionals use when creating programs and applications, while it continues to introduce essential concepts in computer science and coding. The lesson also introduces the idea that computer science can be more than just innovation and creative expression; it can be powerful in trying to solve many problems in today's world. Students apply an Agile development process and task decomposition to solve a problem that meets the needs of others.

Activity 1.2.1 Problem Solving: Interview Database (2 days)
Activity 1.2.2 Algorithms and APIs: Hack Attack (3 days)
Activity 1.2.3 Procedural Abstraction: Price per Slice (3 days)
Activity 1.2.4 Lists: Survey Says (3 days)
Project 1.2.5 App Development: Problem Solving and Innovation (6 days)

12.9.3.MN-QA.4	Employ project management processes using data and tools to deliver quality, value-added products.
CS.9-12.8.1.12.AP.1	Design algorithms to solve computational problems using a combination of original and existing algorithms.
CS.9-12.8.1.12.AP.2	Create generalized computational solutions using collections instead of repeatedly using simple variables.
CS.9-12.8.1.12.AP.3	Select and combine control structures for a specific application based upon performance and readability, and identify trade-offs to justify the choice.
CS.9-12.8.1.12.AP.4	Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue.
CS.9-12.8.1.12.AP.5	Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
CS.9-12.8.1.12.AP.6	Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.
CS.9-12.8.1.12.AP.7	Collaboratively design and develop programs and artifacts for broad audiences by incorporating feedback from users.
CS.9-12.8.1.12.AP.8	Evaluate and refine computational artifacts to make them more usable and accessible.
CS.9-12.8.1.12.AP.9	Collaboratively document and present design decisions in the development of complex programs.
CS.9-12.8.1.12.CS.1	Describe ways in which integrated systems hide underlying implementation details to simplify user experiences.
CS.9-12.8.1.12.CS.4	Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.
CS.9-12.8.1.12.DA.2	Describe the trade-offs in how and where data is organized and stored.

CS.9-12.8.1.12.IC.1	Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices.
CS.9-12.8.1.12.IC.2	Test and refine computational artifacts to reduce bias and equity deficits.
CS.9-12.8.1.12.NI.2	Evaluate security measures to address various common security threats.
CS.9-12.8.1.12.NI.3	Explain how the needs of users and the sensitivity of data determine the level of security implemented.
CS.9-12.8.2.12.ED.5	Evaluate the effectiveness of a product or system based on factors that are related to its requirements, specifications, and constraints (e.g., safety, reliability, economic considerations, quality control, environmental concerns, manufacturability, maintenance and repair, ergonomics).
CS.9-12.8.2.12.NT.1	Explain how different groups can contribute to the overall design of a product.
TECH.8.2.12.B.4	Investigate a technology used in a given period of history, e.g., stone age, industrial revolution or information age, and identify their impact and how they may have changed to meet human needs and wants.
TECH.9.4.12.DC.3	Evaluate the social and economic implications of privacy in the context of safety, law, or ethics (e.g., 6.3.12.HistoryCA.1).

Project 1.2.5

Student Learning Intentions (SLI) WALT: (We are learning to…)	**Project 1.2.5** Decompose a project into smaller parts Apply coding fundamentals and iterative processes Develop an app as part of a Scrum team
Student Learning Strategies	Journaling TEMP Charts (Term, Example, Meaning, Picture) Collaboration Cooperative Learning Pair Programming APB Approach (Activities, Projects, Problems) Class Discussions
Success Criteria	Working app that helps the school manage emergency situations.
Formative Assessment (drives instructional decisions)	Activities and projects Screenshots of code Vocabulary quizzes Call out responses Conclusion questions
Activities and Resources	Create an app that helps your school manage emergency situatio
Suggested Modifications	See Activity 1.1.1

Activity 1.2.4

Student Learning Intentions (SLI) WALT: (We are learning to…)	**Activity 1.2.4** Learn how to create and manipulate lists Learn to use “best so far loops” Learn to use “accumulators” Modify an existing program to be applied to a new purpose Develop an app as part of a pair programming collaboration
Student Learning Strategies	Journaling TEMP Charts (Term, Example, Meaning, Picture) Collaboration Cooperative Learning Pair Programming APB Approach (Activities, Projects, Problems) Class Discussions

Success Criteria	Working Survey Says app
Formative Assessment (drives instructional decisions)	Activities and projects Screenshots of code Vocabulary quizzes Call out responses Conclusion questions
Activities and Resources	Modify an existing app to use a list to store information
Suggested Modifications	See Activity 1.1.1

Activity 1.2.3

Student Learning Intentions (SLI) WALT: (We are learning to…)	**Activity 1.2.3** Create procedural abstractions in a program Learn how to use modulo to make choices in a program Identify what details are hidden or abstracted in block-based programming Develop an app as part of a pair programming collaboration
Student Learning Strategies	Journaling TEMP Charts (Term, Example, Meaning, Picture) Collaboration Cooperative Learning Pair Programming APB Approach (Activities, Projects, Problems) Class Discussions
Success Criteria	Working Price per Slice app
Formative Assessment (drives instructional decisions)	Activities and projects Screenshots of code Vocabulary quizzes Call out responses Conclusion questions
Activities and Resources	Create an app that helps compare various pizza sizes and prices and also calculates how many pizzas to order when feeding a larg group of people.
Suggested Modifications	See Activity 1.1.1

Activity 1.2.2

Student Learning Intentions (SLI) WALT: (We are learning to…)	**Activity 1.2.2** Identify ethical considerations that impact all users Learn how application programming interfaces (API) connect different computing devices Apply algorithms to automate attempts to discover a password Develop an app as part of a pair programming collaboration
Student Learning Strategies	Journaling TEMP Charts (Term, Example, Meaning, Picture) Collaboration Cooperative Learning Pair Programming APB Approach (Activities, Projects, Problems) Class Discussions
Success Criteria	Working Hack Attack app
Formative Assessment (drives instructional decisions)	Activities and projects Screenshots of code Vocabulary quizzes Call out responses Conclusion questions

Activities and Resources	Add to an existing app using APIs and algorithms to automate a brute force attack that will attempt to identify another team's nume password.
Suggested Modifications	See Activity 1.1.1

Activity 1.2.1

Student Learning Intentions (SLI) WALT: (We are learning to…)	**Activity 1.2.1** Learn a problem-solving process Practice communication skills to interview end users of an app Generate ideas for possible problem solving in the future Learn how databases and lists allow data to be collected, persist, and be retrieved Develop an app as part of a pair programming collaboration
Student Learning Strategies	Journaling TEMP Charts (Term, Example, Meaning, Picture) Collaboration Cooperative Learning Pair Programming APB Approach (Activities, Projects, Problems) Class Discussions
Success Criteria	Working Interview Database app
Formative Assessment (drives instructional decisions)	Activities and projects Screenshots of code Vocabulary quizzes Call out responses Conclusion questions
Activities and Resources	Create an app that will sequentially display a list of questions to b asked and capture responses from someone who is surveyed. These interview questions will help identify potential problems to solve.
Suggested Modifications	See Activity 1.1.1

INTERDISCIPLINARY CONNECTIONS: NEW JERSEY STUDENT LEARNING STANDARDS FOR ELA, SOCIAL STUDIES, SCIENCE AND/OR MATHEMATICS

CCSS - English-Language Arts

Key Ideas and Details:

CCSS.ELA-LITERACY.RL.11-12.1 Cite strong and thorough textual evidence to support analysis of what the text says explicitly as well as inferences drawn from the text, including determining where the text leaves matters uncertain.

Integration of Knowledge and Ideas:

CCSS.ELA-LITERACY.W.11-12.1 Write arguments to support claims in an analysis of substantive topics or texts, using valid reasoning and relevant and sufficient evidence.

Production and Distribution of Writing:

CCSS.ELA-LITERACY.W.11-12.4 Produce clear and coherent writing in which the development, organization, and style are appropriate to task, purpose, and audience.

Research to Build and Present Knowledge:

CCSS.ELA-LITERACY.W.11-12.7 Conduct short as well as more sustained research projects to answer a question (including a self-generated question) or solve a problem; narrow or broaden the inquiry when appropriate; synthesize multiple sources on the subject, demonstrating understanding of the subject under investigation.

Range of Writing:

CCSS.ELA-LITERACY.W.11-12.10 Write routinely over extended time frames (time for research, reflection, and revision) and shorter time frames (a single sitting or a day or two) for a range of tasks, purposes, and audiences

CCSS - Mathematics

Reason quantitatively and use units to solve problems:
CCSS.MATH.CONTENT.HSN-Q.A.2 Define appropriate quantities for the purpose of descriptive modeling.
Create equations that describe numbers or relationships:
CCSS.MATH.CONTENT.HSA-CED.A.1 Create equations and inequalities in one variable and use them to solve problems.
Analyze functions using different representations:
CCSS.MATH.CONTENT.HSF-IF.C.7 Graph functions expressed symbolically and show key features of the graph.
Apply geometric concepts in modeling situations:
CCSS.MATH.CONTENT.HSG-MG.A.1 Use geometric shapes, their measures, and their properties to describe objects
Calculate expected values and use them to solve problems:
CCSS.MATH.CONTENT.HSS-MD.A.1 Define a random variable for a quantity of interest by assigning a numerical value to each event in a sample space;
CCSS.MATH.CONTENT.HSS-MD.A.2 Calculate the expected value of a random variable;

English Language Arts

- Journal writing
- Close reading of industry-related content
- Create a brochure for a specific industry
- Keep a running word wall of industry vocabulary

Social Studies

- Research the history of a given industry/profession
- Research prominent historical individuals in a given industry/profession
- Use historical references to solve problems

World Language

- Translate industry-content
- Create a translated index of industry vocabulary
- Generate a translated list of words and phrases related to information technology

Math

- Compare and contrast use of equations and variables in algebra and programming.
- Program graphics and use the properties of geometric shapes
- Compare the computer graphic coordinate system with the Cartesian coordinate plane in math
- Compare probability and the use of random numbers in computer programming.
- Track and track various data, such as industry’s impact on the GDP, career opportunities or among of individuals currently occupying careers

Fine & Performing Arts

- Create a poster recruiting young people to focus their studies on a career in Information Technology

Science

- Research the environmental impact of a given career or industry
- Research latest developments in Information technology
- Investigate applicable-careers in STEM fields